Git- og Composer-kursus

Daniel Schledermann

4. oktober 2016

Indhold

1 Git **2**
1.1 Hvad gør Git? . . . 2
1.1.1 Git er snapshot-baseret. . . . 2
1.1.2 Lege lokalt . . . 2
1.2 Hvordan bruger man git . . . 2
1.2.1 Starte et nyt repo . . . 2
1.2.2 Tilføje nogle filer . . . 3
1.2.3 Oversigt over arbejde . . . 3
1.2.4 Remotes . . . 3
1.2.5 Branching . . . 4
1.2.6 Fortryde . . . 4
1.2.7 Samle ændringerne igen . . . 4
1.2.8 Merge . . . 5
1.2.9 Rebase . . . 5
1.2.10 Ignore-filer . . . 6

2 Composer **6**
2.1 Struktur . . . 6
2.2 Kommandoer . . . 8
2.2.1 Install . . . 9
2.2.2 Update . . . 9
2.2.3 Require . . . 9

3 AU anvendelsen **9**
3.1 Site-repoet . . . 9
3.1.1 At bygge et site: . . . 10
3.1.2 Grundbestandele . . . 10
3.1.3 Installation og afinstallation af extensions . . . 11

3.2 Extension repoer 11
3.3 Fremtiden? 11

Versionsstyring, afhængighedskontrol og teknisk workflow for TYPO3-sites med fokus på AU's miljø.

1 Git

Startet af Linus Torvalds for med formål at tjene som versionsstyring for selve Linux-kernen. Det er fra starten designet til at styre store projekter.

1.1 Hvad gør Git?

Git er et lokal versionsstyring hvor hele datastrukturen; commits, tags, branches ligger lokalt. Det er ikke muligt skærme specifikke grene.

1.1.1 Git er snapshot-baseret.

Det vil vil sige at et commit **ikke** består at forrigge commit + nogle patches som i eksempelvis Subversion eller CVS. I stedet er et commit en erklæring af at **sådan ser filerne ud lige nu**, uanset hvad der ellers findes af commits. Det vil sige at hvis man beskærer et git-træ, så vil et givet commit stadig indehold de samme filer. Det vil også sige at diff's beregnes on-the-fly.

Det har vist sig at være meget effektivt på pladskravet til git. Et stort repo fylder gerne 1/20 af det tilsvarende i Subversion.

1.1.2 Lege lokalt

Fordi alt ligger lokalt kan man også rode frit rundt med sine branches og commits så længe man ikke har delt dem med nogen.

1.2 Hvordan bruger man git

1.2.1 Starte et nyt repo

Det er simpelt. Man behøver ikke nogen server eller andet for at komme i gang.

```
git init
```

Det så sådan set det. Der ligger nu en tom git-datastruktur i ".git"-mappen.

1.2.2 Tilføje nogle filer

Et repo er selvsagt ikke meget værd hvis det ikke indeholder nogle filer. Tekstfiler af en eller anden art håndterer git bedst, men det tager gladelige alle typer af filer.

```
git add enfil.txt
```

Se hvad der er ændret:

```
git status
```

Når man er klar skal man commit'e sit arbejde.

```
git commit -m '[TASK] Did an awesome change'
```

I TYPO3-sammenhæng bruger vi specifikke formater til at angive typen af ændring: https://wiki.typo3.org/CommitMessage_Format_(Git) Dette kan bruges til at danne oversigter over typer af ændringer.

1.2.3 Oversigt over arbejde

Der findes forskellige værktøjer, nogle få:

- git log - giver en shell baseret oversigt
- gitk - giver et X11-vindue
- tig - giver shell baseret interaktivt træ

1.2.4 Remotes

En remote er en forbindelse til et server. Oprettes automatisk hvis man kloner et repo.

```
git clone ssh://gitolite3@au.lnk.lfac.dk/TYPO3CMS/Extensions/aufluidpages
```

Ellers kan man sætte det hvis man har oprettet repoet lokalt:

```
git remote add origin ssh://gitolite3@au.lnk.lfac.dk/TYPO3CMS/Extensions/ausomeext
git push --set-upstream origin master
```

Man taler med en remote med kommandoerne:

- fetch - henter datastrukturer fra remoten
- pull - henter datastrukturer fra remoten og merger ændringer til lokal
- push - sender datastrukturer til remoten

1.2.5 Branching

Dette er nærmest noget af det vigtigste ved git. Det er muligheden for at skrive en helt ny variation af repoet

```
# Hvor står jeg?
git branch -la

# Lav en ny branch herfra
git branch feature/JAHD-666

# Arbejde på den
git checkout feature/JAHD-666
```

Historikken for master og feature/JAHD-666 kan nu udvikle sig frit.

1.2.6 Fortryde

Git kan også nulstille aktuelle ændringer. Smid det hele væk:

```
git reset [--hard]
```

Eller nulstil enkelte filer med:

```
git checkout -- sti/til/fil.txt
```

Hvis man har en smart editor kan den gøre det endnu mere finmasket.

1.2.7 Samle ændringerne igen

Git har forskellige måde at samle ændringer på igen. Blot det mest anvendte måder her.

1.2.8 Merge

Det mest oplagte er at bruge en merge. Dette sker enten eksplicit ved en:

```
git checkout master
git merget feature/JAHD-666
```

Eller det kan ske implicit fra en upstream.

```
git pull
```

Merge beholder den fulde historik over hvad der er sket og viser med et merge-commit hvor sammenfletningen er sket. Dette er normalt enkelt, simpelt og nemt at overskue.

1.2.9 Rebase

PAS PÅ. Med rebase kan man omskrive historikken for at få den simplere. Hvis man har arbejdet rodet med mange lokale branches, kode der har ligget og flydt i lang tid, eller bare fragmenterede commits, så kan man overveje rebase. Rebase må **kun** udføres på egne lokale commit der **ikke er skubbet til en upstream**. Hvornår er det så rimeligt at bruge rebase?

Har man en triviel ændring, men upstream har ændret sig?

```
git pull --rebase
```

Er en lokal feature branch blevet gammel, måske fordi ændringen ikke er godkendt.

```
git pull
git checkout feature/JAHD-666
git rebase master
```

Nu vil feature/JAHD-666 blive "omplantet" til de commits der er sket i mellemtiden på master.

Man har keglet rundt med en masse commits og gerne vil smelte nogle af dem sammen, så det hele fremstår lidt mere læsbart.

```
git rebase -i HEAD~2
```

Og igen: **KUN** på ting der ikke er push'et til nogen remote.

1.2.10 Ignore-filer

De fleste repoer indeholder .gitignore-filer. Disse filer beskriver hvad der specifikt **ikke** skal versionsstyres. Dette er nok så vigtigt. Normalt så ligger afhængigheder, midlertidige filer og brugerindhold i dette filter.

2 Composer

Composer er grundlæggende en dependency manager til PHP. Det har udviklet sig til defacto-standard for PHP. https://getcomposer.org. Derfor har man ved TYPO3 også besluttet sig for at bruge den. Den erstatter eller supplerer en stor del af det arbejde der normalt sker i extension manageren. Vi bruger Composer til at vedligeholde installationerne, både TYPO3-core og extensions.

2.1 Struktur

Et composer-projekt indeholder altid en composer.json-fil. Composer.json identificerer projektet. Et eksempel fra aufluidpages:

```
{
    "name": "aarhus-universitet/aufluidpages",
    "description": "AU templates and backend layouts",
    "type": "typo3-cms-extension",
    "version": "1.7.2"
}
```

Der er et vendor-name, et projektnavn og en version. For et simpelt projekt vil det ofte være det. Det er det for alle vores extensions. For sites, der har en masse afhængigheder, ser det lidt anderledes ud. Her er dependencies defineret med mønster, der kan være defineret composer-repo'er og eventuelt parametre for hentning. Dette udtræk fra i29's composer.json:

```
{
 "minimum-stability": "dev",
 "prefer-stable": true,
 "name": "aarhus-universitet/typo3-i29",
 "description": "Aarhus Universitet TYPO3 i29",
 "repositories": [
  {
   "type": "composer",
```

```
   "url": "https://composer:vah1ea5Ji0op1Pee1xoo@composer-repo.au.dk/"
  },
  {
   "type": "composer",
   "url": "https://composer.typo3.org/"
  }
 ],
 "config": {
  "vendor-dir": "Packages/Libraries",
  "bin-dir": "bin",
  "secure-http": false
 },
 "require": {
  "typo3/cms": "6.2.25",
  "aarhus-universitet/au": "dev-master",
  .....,
  "aarhus-universitet/validateurls": "dev-master",
  "dschledermann/linkservice": "dev-master",
  .....,
  "typo3-ter/datafilter": "*",
  .....,
  "typo3-ter/powermail": "2.17.*",
  .....,
  "typo3-ter/seo-basics": "^0.9.2"
 }
}
```

Ovenstående mønstre vil dog skifte fra tid til anden. Derfor skal man altid beregne konkrete pakker med composer update. Mere om det lidt senere. Disse ændringer skrives ned i composer.lock-filen. Eksempelvis er Powermail beregnet til denne version:

```
{
    "name": "typo3-ter/powermail",
    "version": "2.17.2",
    "dist": {
        "type": "t3x",
        "url": "https://typo3.org/extensions/repository/download/powermail/2.17.2/t3x/
        "reference": null,
        "shasum": null
```

```
    },
    "require": {
        "php": ">= 5.3.0",
        "typo3/cms-core": ">= 6.2.7, <= 7.99.99"
    },
    "replace": {
        "powermail": "self.version"
    },
    "type": "typo3-cms-extension",
    "autoload": {
        "classmap": [
            ""
        ],
        "exclude-from-classmap": [
            "Tests",
            "tests",
            "class.ext_update.php"
        ]
    },
    "authors": [
        {
            "name": "Powermail Development Team",
            "email": "alexander.kellner@in2code.de",
            "company": "in2code.de",
            "username": "wunschtacho"
        }
    ],
    "description": "Powermail is a well-known, editor-friendly, powerful\n        and e
    "time": "2015-12-16 14:40:57"
},
```

Både composer.json og composer.lock skal versionsstyres for at være sikker på at reproducere nøjagtig samme resultat alle steder filerne er til rådighed.

2.2 Kommandoer

Når vi bruger Composer i det daglige arbejde er der normalt tre kommandoer vi bruger

2.2.1 Install

Med composer install bruger man til at bygge projektet præcist som det er beskrevet i composer.lock filen. For en systemadministor vil det normalt være eneste composer-kommando man under daglige omstændigheder kan forvente at skulle bruge.

```
composer install
```

2.2.2 Update

Med composer update kan man ændre på de beregnede pakker. Dette vil normalt være en udvikleropgave. Man kan vælge at fremrykke alle pakker

```
composer update
```

Eller bare nogle specifikke

```
composer update typo3-ter/powermail aarhus-universitet/aufluidpages
```

Dette vil introducere ændringer i composer.lock, som skal commit'es til git for at reproducere ændringen andre steder med composer install.

2.2.3 Require

Dette kan man bruge til at tilføje nye afhængigheder til projektet. Det kan være at man vil tilføje en ny extension

```
composer require typo3-ter/tcdirectmail 3.1.*
```

Dette vil opdatere både composer.json og composer.lock.

3 AU anvendelsen

3.1 Site-repoet

Alle installationer versionsstyres for tiden i det såkaldte "site-repo". Det har adressen: ssh://gitolite3@au.lnk.lfac.dk/Config/site.git

3.1.1 At bygge et site:

Hvis man skal have fat i en installation, i29 til eksempel, vil rækkefølgen være:

```
git clone ssh://gitolite3@au.lnk.lfac.dk/Config/site.git http
cd http
git checkout i29
composer install
```

3.1.2 Grundbestandele

Alle installationer består af overraskende få komponenter:

- .gitignore
- .htaccess
- composer.json
- composer.lock
- robots.txt
- typo3conf/AdditionalConfiguration.example.php
- typo3conf/Basedomain.example.php
- typo3conf/LocalConfiguration.php
- typo3conf/PackageStates.php
- typo3conf/realurl_config.php

Første gang et site opsættes skal typo3conf/AdditionalConfiguration.example.php kopieres over til typo3conf/AdditionalConfiguration.php og værdierne tilpasses den aktuelle installation. Tilsvarende skal typo3conf/Basedomain.example.php også kopieres over til typo3conf/Basedomain.php og tilpasses.

Database og brugerfiler skal indhentes typisk fra backup'en. Thomas&Ole hjælper :-).

3.1.3 Installation og afinstallation af extensions

Skal man tilføje eller fjerne en extension skal man både bruge composer og TYPO3-backenden. Rækkefølgen vil være at man først bruger composer til at "require" extension. Dernæst går man i TYPO3 extension manager og installerer på normal vis. Når man er færdig vil man skulle commit'e følgende

```
git add composer.lock composer.json
git add typo3conf/PackageStates.php typo3conf/LocalConfiguration.php
git commit -m '[FEATURE] Installed crazyext'
git push
```

På demo eller live vil man så kunne køre:

```
git pull
composer install
cd typo3temp
rm -rf Cache/
```

Nu vil extension så både være installeret og konfigureret.

3.2 Extension repoer

Vi har en git-server på:

ssh://gitolite3@au.lnk.lfac.dk/TYPO3CMS/Extensions/<ext name>

Det tager desværre temmelig lang tid at scanne en masse git-repoer. Derfor er der opsat et såkaldt "Satis"-site, der scanner git-repoerne og samler informationerne til et composer-repo. Dette site kan ses her:

https://composer:vah1ea5Ji0op1Pee1xoo@composer-repo.au.dk/

Hvis der tilføjes helt nye git-repoer, så skal de sættes op på "Satis"-sitet.

3.3 Fremtiden?

Med alt under Git- og Composer-kontrol er vejen banet for mere automatiserede processer. Hvilke ønsker er der? Vi tænker naturligvis på continuous integration, eksempelvis med Jenkins.

https://jenkins.io/